Journal of Applied And Basic Machine Intelligence Research (ABMIR)

پژوهش‌های نظری و کاربردی هوش ماشینی

سال چهارم، شماره اول، بهار و تابستان ۱۴۰۵، صفحات ۱۲۸ - ۱۱۳

پیاده‌سازی نسخه کم‌عمق شبکه عصبی کانولوشنی سبک‌وزن SqueezeNet بر روی پلتفرم Versal برای کاربردهای رایانش لبه

زهرا قاسمی[۱]، عاطفه سلیمی شهرکی[۱*]، مهدی عباسی[۲و۳]، محمد لعلی دستجردی[۱]

[۱] گروه مهندسی برق، واحد اصفهان (خوراسگان)، دانشگاه آزاد اسلامی، اصفهان، ایران

[۲] گروه مهندسی کامپیوتر، دانشکده مهندسی، دانشگاه بوعلی سینا، همدان، ایران

[۳] پژوهشکده دانش‌های بنیادی، پژوهشکده علوم کامپیوتر، تهران، ایران

مقاله پژوهشی

چکیده

در سال‌های اخیر، شبکه‌های عصبی عمیق به‌طور گسترده در سامانه‌های بینایی رایانه مورد استفاده قرار گرفته‌اند. با اینحال، پیاده‌سازی این شبکه‌ها برروی وسایل قابل‌حمل به‌دلیل محدودیت‌های منابع محاسباتی، ظرفیت حافظه و توان مصرفی با چالش‌های جدی مواجه است. یکی از راهکارهای مؤثر برای غلبه بر این محدودیت‌ها، بهره‌گیری از پلتفرم‌های پیشرفته ناهمگن مانند AMD Versal است. در این پژوهش، پیاده‌سازی یک نسخه کم‌عمق از شبکه SqueezeNet (حاوی چهار ماژول Fire) برروی تراشه XCVE2302 از پلتفرم AMD Versal VD100 مورد بررسی قرار گرفته است. ابتدا، شبکه با استفاده از کتابخانه PyTorch در محیط Google Colab پیاده‌سازی و با مجموعه‌داده CIFAR-10 آموزش داده شد. سپس، معماری شبکه به‌همراه وزن‌های آموزش‌دیده با استفاده از ابزار Vitis HLS در نرم‌افزار Vivado 2023 به‌صورت سخت‌افزاری پیاده‌سازی گردید. پیاده‌سازی نهایی با فرکانس کاری ۱۰۰ مگاهرتز انجام شد که منجر به استفاده از ۱۲۹ بلوک حافظه BRAM، 136 بلوک حافظه URAM و توان مصرفی حدود ۳/۹۸۴ وات گردید. نتایج به‌دست‌آمده نشان می‌دهد که با طراحی شبکه‌های سبک‌وزن و مدیریت آگاهانه منابع حافظه می‌توان شتاب‌دهنده‌های کارآمدی را برروی پلتفرم Versal پیاده‌سازی کرد. در ادامه این مسیر، بهره‌برداری حداکثری از قابلیت‌های معماری Versal می‌تواند نقش مؤثری در کاهش چالش‌های پیاده‌سازی شبکه‌های عصبی برروی شتاب‌دهنده‌های سخت‌افزاری ایفا کند.

تاریخ دریافت:
۱۴۰۴/۰۹/۰۵

تاریخ پذیرش:
۱۴۰۴/۱۱/۱۲

کلیدواژه‌ها:
شبکه عصبی کانولوشنی، SqueezeNet، Versal، ACAP، شتاب‌دهنده FPGA، هوش مصنوعی لبه‌ای

نویسنده‌مسئول:
Atefehsalimi@iau.ac.ir

doi : 10.22034/ABMIR.2026.24030.1191

E-ISSN: 2821-2037

۱- مقدمه

شبکه‌های عصبی عمیق (DNN)[1] به‌عنوان یکی از مهم‌ترین شاخه‌های هوش مصنوعی (AI)[2] تحول بزرگی در توسعه سیستم‌های بینایی رایانه‌ای[3] [۱] ایجاد کرده‌اند [۲]. با این‌حال، اجرای این شبکه‌ها برروی دستگاه‌های لبه‌ای و قابل حمل[4] به‌دلیل حجم بالای محاسبات آن‌ها همواره با چالش‌های جدی مواجه بوده است. بنابراین، برای رفع این چالش‌ها توجه پژوهشگران به استفاده از شبکه‌های سبک‌وزن معطوف شده است [۳]. شبکه عصبی SqueezeNet [٤] یکی از پیشگامان این حوزه بود و با داشتن تعداد پارامترهای کم و دقت قابل قبول در پیاده‌سازی‌های سخت‌افزاری[5] استفاده شد [۵]. شبکه‌های عصبی معمولاً برروی پردازنده‌های مرکزی (CPU)[6] [٦] و یا گرافیکی (GPU)[7] [۷] پیاده‌سازی می‌شوند. آرایه‌های دروازه‌ای قابل برنامه‌ریزی میدانی (FPGA)[8] نیز به‌دلیل قابلیت موازی‌سازی و مصرف توان پایین گزینه مناسبی برای شتاب‌دهی این شبکه‌ها در سیستم‌های نهفته[9] هستند [۸, ۹]. پلتفرم Versal VD100 [۱۰] عضو خانواده Versal AI Edge شرکت AMD است و با پشتیبانی از حافظه‌های بلوکی (BRAM)[10]، بسیار بزرگ (URAM)[11]، هسته‌های اختصاصی AI و پردازنده‌های ARM چندهسته‌ای در یک تراشه برای کاربردهایی نظیر هوش مصنوعی لبه‌ای[12] با توان پایین به‌کار می‌رود.

در این پژوهش، برای پیاده‌سازی شبکه SqueezeNet برروی تراشه XCVE2302 پلتفرم Versal VD100 ابتدا از شبکه اصلی (دارای هشت ماژول Fire) استفاده شده است. به‌دلیل مصرف بالای منابع حافظه این پیاده‌سازی با شکست مواجه شد. در ادامه، تعداد ماژول‌های Fire شبکه SqueezeNet از هشت به چهار کاهش یافت تا عمق شبکه و حجم پارامترهای آن کاهش یابد. در این‌حالت میزان استفاده از منابع حافظه کاهش یافت و پیاده‌سازی موفق بود. این نسخه از شبکه SqueezeNet با استفاده از مجموعه داده CIFAR-10 [۱۱] آموزش دیده است. نتایج پیاده‌سازی نشان می‌دهد که کاهش عمق شبکه منجر به کاهش مصرف منابع حافظه و تأخیر استنتاج نسبت به نسخه اصلی همان شبکه شده است.

نوآوری مربوط به این پژوهش صرفاً در کاهش عمق شبکه SqueezeNet خلاصه نمی‌شود، بلکه در ارائه یک راهبرد هم‌طراحی[13] شبکه عصبی و ساختار سلسله‌مراتبی حافظه در پلتفرم Versalنیز نهفته است. در این رویکرد، معماری شبکه براساس محدودیت‌ها و قابلیت‌های حافظه‌های داخلی BRAM و URAMتعیین شده و تصمیمات معماری به‌صورت یک سخت‌افزارآگاه[14] اتخاذ شده‌اند. لازم به‌ذکر است شتاب‌دهنده این پژوهش صرفاً در بخش منطق برنامه‌پذیر (PL)[15] پیاده‌سازی شده و در نسخه حاضر از قابلیت‌های خاص AI Engine، NoC و DPU استفاده نشده است. هدف، ارائه یک پیاده‌سازی قابل کنترل و قابل تحلیل در سطح HLS و مبتنی‌بر مدیریت سلسله‌مراتبی حافظه BRAM/URAM است.

در ادامه، بخش‌های مختلف این پژوهش ذکر می‌شوند. بخش ۲ به‌مروری برکارهای مرتبط اختصاص دارد. بخش ۳ روش مورد استفاده در این پژوهش را تشریح می‌کند. بخش ٤ به‌معرفی پلتفرم Versal VD100 می‌پردازد. بخش ۵ جزئیات پیاده‌سازی و نتایج آن‌را ارائه می‌دهد. بخش ٦ نیز نتایج به‌دست‌آمده را تحلیل و بررسی می‌کند. نهایت بخش ۷ به نتیجه‌گیری و پیشنهادهایی برای تحقیقات آتی می‌پردازد.

[1] Deep Neural Networks (DNN)
[2] Artificial Intelligence (AI)
[3] Computer Vision
[4] Edge and portable devices
[5] Hardware Implementation
[6] Central Processing Unit (CPU)
[7] Graphics Processing Unit (GPU)
[8] Field Programmable Gate Arrays (FPGAs)
[9] Embedded System
[10] Block RAM (BRAM)
[11] Ultra RAM (URAM)
[12] Artificial Intelligence (AI) Edge
[13] Co-design
[14] Hardware-Aware
[15] Programmable Logic (PL)

۲- پیشینه پژوهش

پیاده‌سازی شبکه‌های عصبی کانولوشنی (CNN)[1] برروی شتاب‌دهنده‌های FPGA با چالش‌هایی مانند محدودیت‌های منابع حافظه و بلوک‌های پردازش سیگنال دیجیتال (DSP)[2] روبه‌رو است [۱۲-۱۴]. پژوهشگران برای غلبه‌براین محدودیت‌ها دو رویکرد اصلی یعنی طراحی شتاب‌دهنده سخت‌افزاری بهینه با بهره‌گیری از منابع موجود و یا فشرده‌سازی شبکه‌های عصبی عمیق (با کاهش تعداد پارامترها و عملیات محاسباتی) را دنبال می‌کنند [۱۴, ۱۵]. برای طراحی شتاب‌دهنده‌های سخت‌افزاری، پژوهش‌های متعددی بر بهبود کارایی، توان و بهره‌وری منابع متمرکز بوده‌اند. وینیا و همکاران [۱۶] ابزاری مبتنی‌بر کتابخانه شبکه عصبی سریع (FANN)[3] برای استخراج پارامترهای شبکه و تسهیل پیاده‌سازی سخت‌افزاری بر روی FPGA ارائه کردند که موجب کاهش زمان طراحی شد. لی و همکاران [۱۷] با استفاده از معماری شتاب‌دهنده کاملاً خطلوله‌ای[4] و حذف بافرهای میان‌لایه‌ای، گذردهی داده[5] و بهره‌وری منابع را به‌طور قابل توجهی افزایش دادند. کیم و همکاران [۱۸] با طراحی و بهینه‌سازی یک واحد محاسبات ترکیبی (MAC)[6] در سطح انتقال رجیستر (RTL)[7] توان عملیاتی شتاب‌دهنده FPGA را بهبود بخشیدند. اسلدویچ و همکاران [۱۹] با کمک محاسبات نقطه ثابت[8] ۱۶ بیتی و دودویی‌سازی هسته‌ها[9] منابع مورد استفاده در یک سیستم مبتنی‌بر FPGA را کاهش دادند. برای فشرده‌سازی شبکه‌های عصبی عمیق نیز پژوهش‌های متعددی انجام شده است. این پژوهش‌ها باعث طراحی و توسعه شبکه‌های سبک‌وزن مانند SqueezeNet [۴]، MobileNet [۲۰]، ShuffleNet [۲۱] و سایر شبکه‌های مشابه شده‌اند. شبکه SqueezeNet [۴] یکی از اولین شبکه‌های سبک‌وزن است. گشوند و همکاران [۲۲] شتاب‌دهنده ZynqNet مبتنی‌بر سنتز سطح بالا (HLS)[10] را به‌عنوان یک نسخه بهینه‌شده SqueezeNet برای استفاده در پلتفرم‌های مبتنی‌بر FPGA کم‌مصرف (مانند Zynq [۲۳]) معرفی کردند. این مدل با استفاده از ساختار منظم و حلقه‌های تودرتو میزان دسترسی به‌حافظه را به‌حداقل رساند و باوجود محدودیت منابع در پلتفرم XC-7Z045 سری Zynq-7000 [۲۳] توانست عملکرد خوبی را ارائه دهد. موسولیوتیس و همکاران [۲۴] نیز برای پیاده‌سازی شبکه SqueezeNet برروی پلتفرم Zynq یک شتاب‌دهنده مبتنی‌بر HLS به‌نام SqueezeJet را معرفی کردند. در این‌کار، لایه‌های کانولوشنی شبکه به‌عنوان یک شتاب‌دهنده مشترک و جریان داده‌محور[11] طراحی شدند تا میزان وابستگی به حافظه خارجی کاهش یابد. سپس، پژوهشگران مذکور [۲۵] شتاب‌دهنده SqueezeJet-3 را معرفی کردند. این شتاب‌دهنده با تمرکز بر اجرای شبکه‌های کوچک از جمله SqueezeNet و ZynqNet برای استفاده در پلتفرم‌های Zynq طراحی شده است. معماری SqueezeJet-3 از طریق محاسبات نقطه ثابت هشت بیتی و بهینه‌سازی فیلترهای ۱×۱ و ۳×۳ امکان پیاده‌سازی کارآمد شبکه‌های کانولوشنی برروی این نوع پلتفرم‌ها را فراهم می‌کند. شتاب‌دهنده‌های SqueezeJet و SqueezeJet-3 برروی پلتفرم XC-7Z020 سری Zynq-7000 پیاده‌سازی شدند و به‌نتایج قابل قبولی دست یافتند. سایر شبکه‌های سبک‌وزن نیز می‌توانند برروی پلتفرم‌های Zynq پیاده‌سازی شوند. ژائو و همکاران [۲۶] شبکه‌های MobileNetV2 و ShuffleNetV2 را برروی پلتفرم Xilinx ZC706 پیاده‌سازی کردند تا شتاب‌دهنده مقیاس‌پذیر خود را ارزیابی کنند. نتایج ارزیابی آن‌ها نشان داد که استفاده از شبکه‌های سبک‌وزن به‌همراه یک معماری جریانیِ مقیاس‌پذیر برای شتاب‌دهنده[12] می‌تواند میزان دسترسی به حافظه خارجی و مصرف منابع پلتفرم را کاهش دهد. ژو و همکاران [۲۷] نیز برای پیاده‌سازی شبکه MobileNetV3 برروی پلتفرم Zynq از یک روش

[1] Convolutional Neural Network (CNN)
[2] Digital Signal Processing (DSP)
[3] Fast Artificial Neural Network (FANN)
[4] Fully Pipelined
[5] Data Throughput
[6] Multiply-Accumulate (MAC)
[7] Register-Transfer Level (RTL)
[8] Fixed-Point
[9] Kernel Binarization
[10] High-Level Synthesis (HLS)
[11] Data-driven flow
[12] A Scalable Streaming Accelerator Architecture

کوانتیزاسیون دینامیک[1] همراه با شتاب‌دهی FPGA استفاده کردند. این روش نشان داد که توانایی حفظ دقت مدل و کاهش مصرف منابع و زمان استنتاج را دارد. به‌طورمشابه، سایر شبکه‌های سبک‌وزن مانند YOLO، ResNet، EfficientNet و VGG با استفاده از انواع پلتفرم‌های Zynq پیاده‌سازی شده‌اند و به‌نحوی عملکرد شتاب‌دهنده را بهینه کرده‌اند [۲۸-۳۰].

پلتفرم‌های پیشرفته AMD Versal AI Edge [۳۱] با پشتیبانی از IP قابل پیکربندی به‌نام واحد پردازش یادگیری عمیق (DPU)[2] و ابزار Vitis AI امکان پیاده‌سازی سریع شبکه‌های عصبی را فراهم می‌کنند. درنتیجه، این پلتفرم‌های کم‌حجم و کم‌مصرف می‌توانند در کاربردهای خاص مانند رانندگی خودکار استفاده شوند [۳۲]. تعدادی از پژوهشگران برای شتاب‌دهی شبکه‌های عصبی از پلتفرم‌های Versal استفاده کردند. الزوبی و همکاران [۳۳] یک شبکه عصبی کانولوشنی سفارشی را برروی پلتفرم Versal پیاده‌سازی کردند. نتایج این پیاده‌سازی نشان داد که توان عملیاتی و تأخیر این پلتفرم در مقایسه با CPU، GPU و نسل‌های قبلی FPGA بهبود قابل توجهی پیدا کرده است. ژانگ و همکاران [۳٤] چارچوب شتاب‌دهنده ترنسفور (CAT)[3] را برروی پلتفرم Versal پیاده‌سازی کردند و نشان دادند که این چارچوب با بهره‌گیری از قابلیت‌های ناهمگن Versal می‌تواند در مقایسه با GPU و FPGA بهبود قابل‌توجهی در توان عملیاتی و بهره‌وری انرژی برای مدل‌های ترنسفورمر فراهم کند. پریمن و همکاران [۱۱] معماری شتاب‌دهنده Versal را برای پردازش داده‌ها در سیستم‌های فضایی ارزیابی کردند. نتایج نشان داد که زیربخش‌های ناهمگن و منطق برنامه‌پذیر پلتفرم Versal تعادل مناسبی بین کارایی و مصرف انرژی آن فراهم می‌کند. به‌طورکلی، با پیاده‌سازی شبکه‌های عصبی برروی نسل‌های جدید پلتفرم‌های AMD Versal AI Edge و طراحی شتاب‌دهنده‌های پیشرفته می‌توان از آن‌ها در کاربردهایی مانند بینایی [۳۵, ۳٦]، رمزنگاری هوشمند شبکه [۳۷]، سیستم‌های فضایی [۳۸] و سایر حوزه‌های هوشمند استفاده کرد و ضمن افزایش کارایی سیستم‌ها، انرژی مصرفی آن‌ها را به‌حداقل رساند.

پلتفرم‌های Versal(به‌ویژه VD100) امکانات و قابلیت‌های گسترده‌ای را برای پیاده‌سازی شبکه‌های عصبی فراهم می‌کنند. در این پژوهش، با الهام از شتاب‌دهنده ZynqNet [۲۲]، یک نسخه کم‌عمق از شبکه SqueezeNet (حاوی ٤ ماژول Fire به‌جای ۸ ماژول مدل اصلی) برروی پلتفرم Versal VD100 پیاده‌سازی شده است. تمرکز این مطالعه برروی استفاده از منابع عمومی FPGA و بررسی عملی تأثیر کاهش عمق شبکه برروی مصرف منابع سخت‌افزاری پلتفرم و تأخیر استنتاج شتاب‌دهنده است و ادعایی درخصوص بهره‌گیری هدفمند از قابلیت‌های اختصاصی پلتفرم Versal یا ارائه معماری شتاب‌دهنده جدید ندارد. ZynqNet از پلتفرم XC-7Z045 که منابع حافظه محدودی دارد، استفاده کرده است. اما، پژوهش حاضر با استفاده همزمان از حافظه‌های BRAM و URAM موجود در پلتفرم Versal VD100 محدودیت‌های منابع حافظه را جبران کرده است.

۳- روش مورد استفاده در این پژوهش

در این پژوهش، نسخه کم‌عمق شبکه SqueezeNet [٤] (حاوی ٤ ماژول Fire) برروی تراشه XCVE2302 پلتفرم Versal VD100 شرکت AMD پیاده‌سازی شده است. در این پژوهش، طراحی معماری شبکه عصبی به‌صورت مستقیم تحت تأثیر ساختار حافظه داخلی پلتفرم Versal انجام شده است. برخلاف روش‌های متداول که معماری شبکه ابتدا تعیین و سپس به سخت‌افزار نگاشت می‌شود در این کار ظرفیت، پهنای باند و الگوی دسترسی همزمان به حافظه‌های BRAM و URAM به‌عنوان قیود اصلی طراحی در نظر گرفته شده‌اند. کاهش تعداد ماژول‌های Fire از ۸ به ٤ نیز نتیجه یک فرآیند سیستماتیک مبتنی‌بر امکان‌پذیری نگاشت وزن‌ها و نقشه‌های ویژگی در حافظه داخلی Versal بوده است. این راهبرد طراحی در پلتفرم‌هایی نظیر Zynq-7000 به‌دلیل فقدان URAMقابل پیاده‌سازی نیست. برای رسیدن به هدف پژوهش ابتدا، شبکه SqueezeNet کم‌عمق با استفاده از PyTorch در فضای ابری Colab پیاده‌سازی و با مجموعه‌داده CIFAR-10 آموزش دیده است. سپس، معماری شبکه به‌همراه وزن‌های

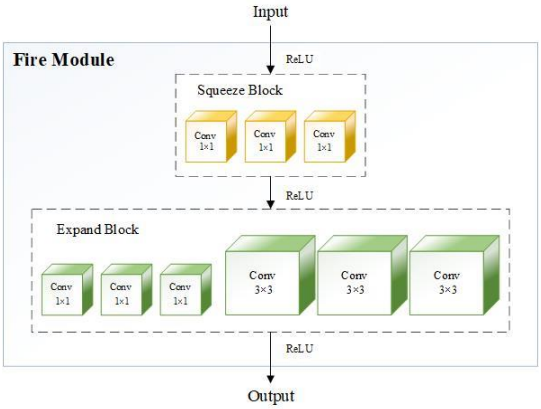
[1] Dynamic Quantization
[2] Deep Learning Processing Unit (DPU)
[3] Transformer Accelerator Framework (CAT)

آموزش‌دیده در ابزار Vitis HLS نرم‌افزار Vivado 2023 پیاده‌سازی می‌شوند. در ادامه، ابتدا معماری شبکه اصلی SqueezeNet و سپس شبکه کم‌عمق مورد استفاده شرح داده خواهند شد.

۳-۱ شبکه SqueezeNet

شبکه SqueezeNet [٤] یک نوع شبکه کانولوشنی است و به‌گونه‌ای طراحی شده که بدون تغییر در دقت شبکه حجم آن‌را کاهش دهد. این شبکه با استفاده از ساختار بهینه‌شده‌ای که دارد می‌تواند برروی سخت‌افزارهای مبتنی‌بر FPGA اجرا شود و ویژگی‌های پیچیده داده‌ها را استخراج کند. شکل (۱) ماژول Fire که هسته اصلی شبکه SqueezeNet است را نشان می‌دهد. این ماژول از بلوک‌های فشرده‌سازی[1] و بسط‌دهنده[2] ساخته شده است. بخش فشرده‌سازی شامل لایه کانولوشنی ۱×۱ است و تعداد کانال‌های ورودی را کاهش می‌دهد. بخش بسط‌دهنده نیز شامل دو شاخه موازی (لایه‌های کانولوشنی ۱×۱ و ۳×۳) است. شاخه ۱×۱ ویژگی‌های سطح بالای داده‌ها و شاخه ۳×۳ جزئیات مکانی آن‌ها را استخراج می‌کند. درنهایت خروجی‌های این دو شاخه موازی در محور کانال ادغام شده و به لایه بعدی انتقال داده می‌شوند. این ساختار باعث می‌شود که شبکه حتی با داشتن تعداد پارامترهای کم نیز بتواند تصاویر را با دقت قابل قبولی تشخیص دهد.

شکل (۲) ساختار شبکه SqueezeNet را نشان می‌دهد. این شبکه شامل لایه کانولوشن ابتدایی (conv1)، هشت ماژول Fire (fire2 تا fire9)، لایه ادغام بزرگترین مقدار[3]، لایه کانولوشن انتهایی (conv10) و لایه‌های میانگین‌گیری عمومی[4] و softmax است. در این ساختار لایه‌های کانولوشن ابتدایی و انتهایی به‌ترتیب مسئول استخراج ویژگی‌های سطح پایین و سطح بالای تصویر ورودی هستند. ماژول‌های Fire نیز وظیفه کاهش حجم پارامترهای شبکه و استخراج ویژگی‌های پیچیده داده‌ها را به‌عهده دارند.



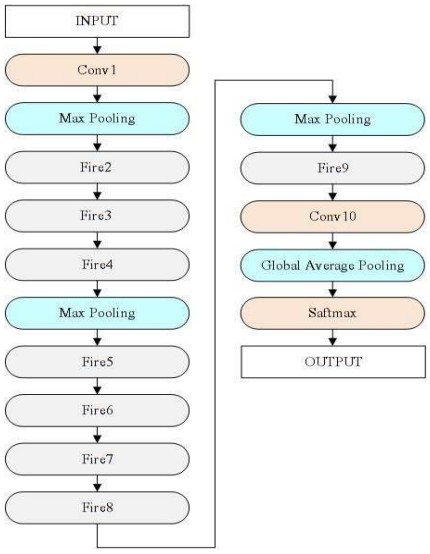
شکل (۱): ساختار داخلی ماژول Fire

شکل (۲) ساختار شبکه SqueezeNet را نشان می‌دهد. این شبکه شامل لایه کانولوشن ابتدایی (conv1)، هشت ماژول Fire (fire2 تا fire9)، لایه ادغام بزرگترین مقدار[5]، لایه کانولوشن انتهایی (conv10) و لایه‌های میانگین‌گیری عمومی[6] و softmax است. در این ساختار لایه‌های کانولوشن ابتدایی و انتهایی به‌ترتیب مسئول استخراج ویژگی‌های سطح پایین و سطح بالای تصویر ورودی هستند. ماژول‌های Fire نیز وظیفه کاهش حجم پارامترهای شبکه و استخراج ویژگی‌های پیچیده داده‌ها را به‌عهده دارند.

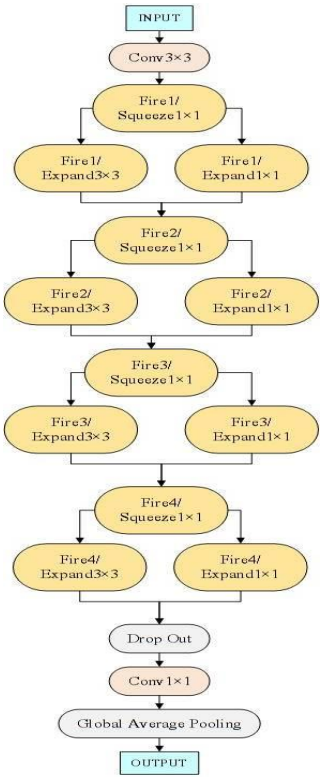
۳-۲ شبکه SqueezeNet کم‌عمق

در شکل (۳) بلوک دیاگرام شبکه squeezenet با استفاده از چهار ماژول Fire نمایش داده شده است. این لایه‌ها پس از لایه کانولوشن ابتدایی قرار گرفته‌اند تا به‌تدریج ویژگی‌های سطح بالاتر داده‌ها را استخراج کنند. نرمال‌سازی دسته‌ای[7] پس از هر لایه کانولوشن باعث پایداری در آموزش شبکه می‌شود. تابع فعال‌سازی ReLU نیز باعث رفتار غیرخطی مدل شده و از کوچک شدن بیش از حد گرادیان[8] (که باعث کند شدن یادگیری می‌شود) جلوگیری می‌کند. درنهایت، لایه Dropout برای جلوگیری از بیش‌برازش[9] و ادغام متوسط سراسری نیز برای تولید بردار ویژگی نهایی پیش از طبقه‌بندی به‌کار رفته‌اند.

[1] Squeeze
[2] Expand
[3] Max Pooling
[4] Global Average Pooling
[5] Max Pooling
[6] Global Average Pooling
[7] Batch Normalization
[8] Vanishing gradients
[9] Overfitting



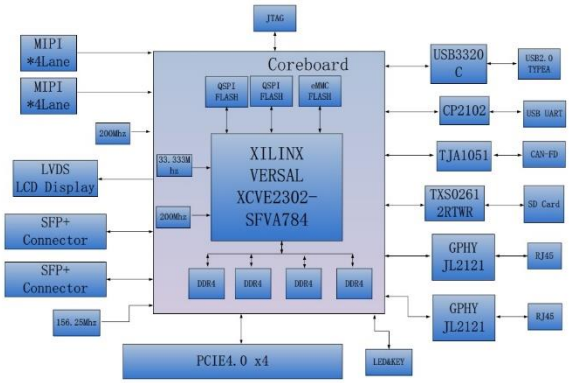
شکل (۲): ساختار پایه شبکه SqueezeNet



شکل (۳): ساختار شبکه SqueezeNet با چهار ماژول Fire

٤- پلتفرم Versal VD100

شکل (٤) بلوک‌های تشکیل‌دهنده پلتفرم Versal VD100 را نشان می‌دهد. این پلتفرم برپایه تراشه XCVE2302 ساخته شده و شامل ترکیب یک ماژول پردازشی (SOM) و یک برد پایه است. روی برد پایه مجموعه‌ای از رابط‌های کاربردی مانند PMOD، USB2.0، UART، PCIe4.0، MIPI، فیبر نوری و اترنت در کنار حافظه‌های DDR4، QSPI و eMMC قرار گرفته‌اند تا امکان ارتباط سریع، پایدار و انعطاف‌پذیر با تجهیزات مختلف را فراهم کنند. به‌طورکلی، این قابلیت‌ها باعث شده که پلتفرم Versal VD100 گزینه مناسبی برای اجرای الگوریتم‌های هوش مصنوعی باشد [۱۰]. لازم به تأکید است که در این پژوهش از AI Engine یا DPU پلتفرم Versal استفاده نشده و تمرکز طراحی بر منطق برنامه‌پذیر و بهره‌گیری از ساختار سلسله‌مراتبی حافظه BRAM/URAMبوده است. این انتخاب با هدف ارائه یک شتاب‌دهنده قابل کنترل و قابل تحلیل در سطح HLS انجام شده است.

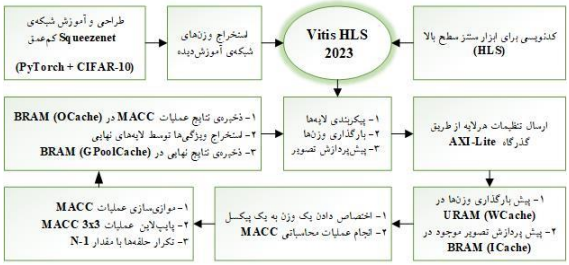


شکل (٤): بلوک‌های داخلی پلتفرم Versal VD100 [۱۰]

۵- نحوه پیاده‌سازی و نتایج آن

شکل (۵) یک نمای کلی از تمامی مراحل اجرای روش مورد استفاده در این پژوهش را نشان می‌دهد. ابتدا، شبکه عصبی SqueezeNet کم‌عمق (شامل چهار ماژول Fire به‌جای هشت ماژول) با استفاده از PyTorch و مجموعه داده CIFAR-10 [۱۱] در محیط Colab آموزش داده می‌شود. سپس، وزن‌های به‌دست آمده استخراج و از طریق ابزار Vitis HLS در حافظه URAM پلتفرم Versal VD100 ذخیره می‌شوند. این‌کار باعث می‌شود که شتاب‌دهنده به‌صورت مستقیم از وزن‌ها استفاده کند و هنگام پردازش به‌صورت مداوم به‌حافظه اصلی مراجعه نکند. درنتیجه، سرعت پردازش شتاب‌دهنده افزایش می‌یابد. در مرحله بعد، هر

تصویر ورودی پس از اعمال پیش‌پردازش‌های لازم با وزن‌های مربوطه ترکیب و ویژگی‌های سطح بالای آن استخراج می‌گردند. نتایج این پردازش‌ها نیز در حافظه‌های داخلی ذخیره می‌شوند تا شتاب‌دهنده بتواند با کمترین تأخیر لایه‌های بعدی را پردازش کند. مشاهده می‌شود که این مراحل در یک حلقه سازمان‌یافته انجام می‌شوند و تا زمانی‌که کل تصویر پردازش نشده باشد این چرخه ادامه می‌یابد. یکی از ویژگی‌های کلیدی این طراحی، انعطاف‌پذیری آن در برابر ساختارهای مختلف شبکه است. به این ترتیب که پیکربندی هر لایه (مانند اندازه فیلتر یا تعداد کانال‌ها) دقیقاً مطابق با معماری اصلی شبکه عصبی تنظیم می‌شود. این رویکرد باعث اجرای کارآمد شبکه عصبی برروی سخت‌افزار اختصاصی می‌شود. در ادامه این مراحل توضیح داده می‌شوند.



شکل (۵): مراحل اجرای روش پیشنهادی این پژوهش

۵-۱ شبکه SqueezeNet کم‌عمق

برای اجرای شبکه SqueezeNet بهینه‌شده ابتدا هسته اصلی آن (ماژول Fire) تعریف می‌شود. این ماژول از بخش‌های فشرده‌سازی و توسعه تشکیل شده است. در بخش فشرده‌سازی از یک لایه کانولوشن با فیلتر ۱×۱ (k=1) برای کاهش تعداد کانال‌های ورودی استفاده می‌شود. بخش توسعه نیز از دو شاخه موازی با لایه‌های کانوشنی با فیلترهای ۱×۱ و ۳×۳ (k=1, k=3) ساخته می‌شود. خروجی بخش فشرده‌سازی به‌صورت مستقل به هرکدام از شاخه‌ها اعمال شده تا به‌صورت هم‌زمان ویژگی‌های محلی و ساختارهای مکانی داده‌ها را استخراج کنند. درنهایت خروجی شاخه‌ها با یکدیگر ترکیب شده و خروجی نهایی ماژول Fire را مشخص می‌کنند. در هر بخش از نرمال‌سازی دسته‌ای و تابع فعال‌سازی ReLU استفاده می‌شود تا با کاهش نوسانات و حذف مقادیر بی‌اهمیت باعث پایدارتر شدن شبکه شوند. استفاده از Inplace=True نیز باعث کاهش مصرف حافظه می‌شود زیرا در هرلحظه ورودی‌های لایه‌ها را جایگزین خروجی‌های آن‌ها می‌کند. جدول (۱) ساختار ماژول Fire را نمایش می‌دهد.

جدول (۱): ساختار ماژول Fire

بخش‌های اصلی ماژول Fire	لایه‌های داخلی هر بخش	توضیحات هر لایه
Squeeze	Conv2d	Input Channel to Squeeze Channel, k=1, Inplace=True
	BatchNorm2d	
	ReLU	
Expand 1x1	Conv2d	Squeeze Channel to Expand 1x1channel, k=1, Inplace=True
	BatchNorm2d	
	ReLU	
Expand 3x3	Conv2d	Squeeze Channel to Expand 3x3 Channel, k=3, padding=1, Inplace=True
	BatchNorm2d	
	ReLU	
Output: Concatenate ([expand1x1, expand3x3], dim=1)		

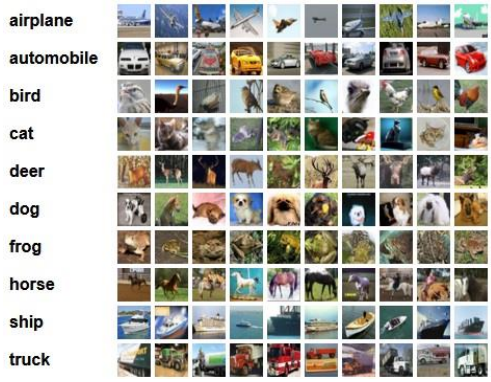
در جدول (۲) ساختار شبکه عصبی SqueezeNet کم‌عمق (حاوی چهار ماژول Fire) استفاده شده در این پژوهش را نشان می‌دهد. تصویری با ابعاد ۳×۳۲×۳۲ به‌لایه کانولوشنی اولیه با ٦٤ فیلتر ۳×۳ اعمال می‌شود. برای پایداری شبکه از نرمال‌سازی دسته‌ای و تابع فعال‌سازی ReLU به‌همراه با Inplace=True استفاده می‌شود. سپس، ماژول‌های Fire ویژگی‌های داده‌ها را استخراج می‌کنند. لایه MaxPool2d نیز وظیفه استخراج ویژگی‌ها و کاهش ابعاد نقشه ویژگی[1] را به‌عهده دارد. از لایه Dropout برای جلوگیری از بیش‌برازش[2] شبکه استفاده می‌شود. لایه کانولوشنی ۱×۱ نهایی با داشتن ۱۰ فیلتر نیز تعداد کانال‌های خروجی شبکه را تنظیم می‌کند. درنهایت، لایه GlobalAvgPool تمامی اطلاعات را به قسمت طبقه‌بندی نهایی شبکه اعمال می‌کند. این ساختار با جایگزینی GlobalAvgPool (به‌جای لایه‌های کاملاً متصل[3]) نه‌تنها حجم مدل را کاهش داده بلکه قابلیت تعمیم‌پذیری شبکه را نیز بهبود می‌بخشد.

[1] Feature Map
[2] Overfitting
[3] Fully Connected

جدول (۲): ساختار شبکه SqueezeNet با چهار ماژول Fire

اسم لایه	تعداد کانال‌های خروجی هر لایه	پارامترهای لایه
Input	3	32×32×3
Conv2d	64	k=3, s=2, p=1
BatchNorm + ReLU	64	-
Fire2	32	Squeeze=16, Expand=16+16
Fire3	64	Squeeze=16, Expand=32+32
MaxPool2d	64	k=3, s=2
Fire4	128	Squeeze=32, Expand=64+64
Fire5	196	Squeeze=32, Expand=98+98
MaxPool2d	196	k=3, s=2
Dropout	196	p=0.6
Conv2d	10	k=1
GlobalAvgPool2d	10	output size = 1
Flatten	10	-

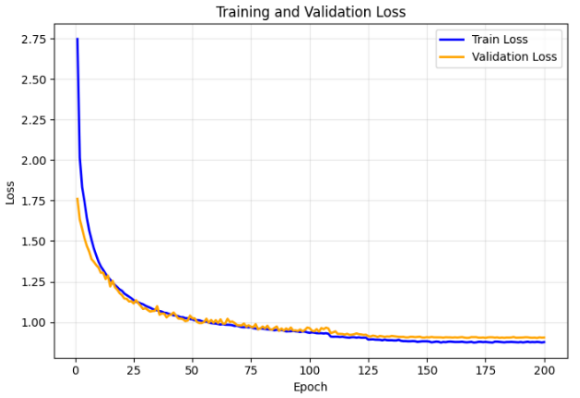
برای آموزش شبکه SqueezeNet کم‌عمق مورد استفاده در این پژوهش از مجموعه داده CIFAR-10 استفاده می‌شود. این مجموعه داده یکی از مجموعه داده‌های استاندارد و پرکاربرد در حوزه یادگیری عمیق به‌حساب آمده و شامل ۶۰۰۰۰ تصویر رنگی (در ۱۰ کلاس) با ابعاد ۳۲×۳۲ پیکسل می‌شود. این مجموعه‌داده به‌دلیل تنوع کلاس‌های تصاویر به‌طور گسترده برای ارزیابی عملکرد شبکه‌های عصبی مختلف مانند SqueezeNet مورد استفاده قرار می‌گیرد. در شکل (۶) نمونه‌هایی از تصاویر موجود در این مجموعه داده به‌همراه نام هر کلاس نشان داده شده است.



شکل (۶): مجموعه داده CIFAR-10 [۱۱]

شکل (۷) منحنی‌های دقت آموزش[1] و ارزیابی[2] شبکه در طول دوره‌های مختلف آموزش را نشان می‌دهد. شبکه پیشنهادی به دقت آموزش ۸۵/۵۶٪ و ارزیابی ۸۴/۷۰٪ دست یافته است. شکل (۸) نیز روند تغییرات تابع زیان در مراحل آموزش[3] و ارزیابی[4] را نشان می‌دهد. این نتایج نشان می‌دهند که شبکه squeezenet کم‌عمق می‌تواند الگوهای مختلف تصاویر را یاد بگیرد و عملکردی پایدار و قابل اعتماد داشته باشد.

شکل (۷): منحنی دقت آموزش شبکه بهینه شده



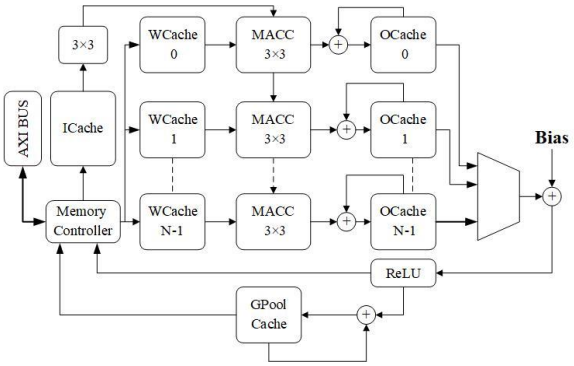
شکل (۸): منحنی عدد زیان شبکه بهینه شده

۵-۲ پیاده‌سازی برروی تراشه XCVE2302

پس از پیاده‌سازی شبکه SqueezeNet کم‌عمق، وزن‌های شبکه آموزش‌دیده استخراج شده و به‌همراه ساختار کلی شبکه به ابزار Vitis HLS 2023 اعمال می‌شوند. این پژوهش با الهام از روش مورد استفاده در ZynqNet [۲۲] عملیات استنتاج شبکه را

[1] Training Accuracy
[2] Validation Accuracy
[3] Training Loss
[4] Validation Loss

به‌صورت کارآمدی اجرا می‌کند. شکل (۹) نحوه سازماندهی اجزای شتاب‌دهنده Versal VD100 اختصاصی پژوهش را نشان می‌دهد. مطابق شکل، موازی‌سازی با استفاده از N واحد پردازشی و حافظه‌های پنهان انجام می‌شود. ابتدا داده‌ها از طریق کنترل‌کننده حافظه و گذرگاه AXI وارد شده و در حافظه پنهان ورودی (ICache)[1] ذخیره می‌شوند. وزن‌ها به‌همراه Bias آن‌ها نیز در حافظه پنهان وزن‌ها (WCache)[2] ذخیره می‌شوند. هرکدام از واحدهای پردازشی با استفاده از بلوک‌های محاسباتی (MACC)[3] محاسبات را به‌صورت موازی انجام داده و نتایج به‌دست آمده را در حافظه‌های پنهان خروجی (OCache)[4] ذخیره می‌کنند. درنهایت، با ادغام خروجی‌های موردنظر در لایه ادغام سراسری (GPool)[5] و اعمال تابع فعال‌سازی ReLU پردازش نهایی انجام می‌شود. این ساختار باعث افزایش سرعت محاسبات شبکه‌های عصبی می‌شود.



شکل (۹): سازماندهی اجزای شتاب‌دهنده Versal VD100

تمامی اجزای شتاب‌دهنده توسط زبان برنامه‌نویسی C/C++ در ابزار Vitis HLS 2023 نوشته شده و پس از اجرای موفقیت‌آمیز عملیات سنتز[6] به‌یک IP واحد مطابق با شکل (۱۰) تبدیل شده‌اند. این IP به‌عنوان هسته اصلی شتاب‌دهنده فعالیت کرده و ارتباط با سایر بلوک‌های پلتفرم مانند واسط‌های استاندارد AXI4-Lite و کنترل AXI4-Streamرا فراهم می‌کند. این واسط‌ها برای تنظیمات و کنترل پلتفرم از سوی پردازنده استفاده می‌شوند. همچنین این IP ارتباط با واسط‌های حافظه و سیگنال وقفه را برقرار کرده تا پلتفرم بتواند با بلوک‌ها و سیستم‌های دیگر تعامل داشته باشد. شکل (۱۱) بلوک‌دیاگرام ترسیم‌شده مربوط به تست IP شتاب‌دهنده (که توسط ابزار Vitis HLS ساخته و بهینه‌سازی شد و در شکل با DUT مشخص شده است) را نشان داده است. سایر بلوک‌ها به‌طور پیش‌فرض در نرم‌افزار Vivado وجود دارند و نحوه ارتباط و راه‌اندازی شتاب‌دهنده را مشخص می‌کنند.

شکل (۱۰): IP شتاب‌دهنده برای شبکه موردنظر

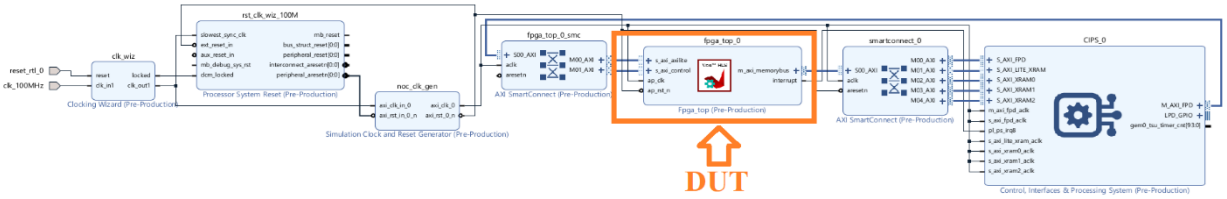
۶- تحلیل و بررسی

در این بخش، نتایج حاصل‌شده از پیاده‌سازی مورد تحلیل و بررسی قرار می‌گیرند. مشخصات شبکه‌های SqueezeNet [۴] اصلی و کم‌عمق در جدول (۳) نشان داده شده است. مشخصات شبکه اصلی طبق اطلاعات موجود در سایت‌های گیت‌هاب[7] و هاگینگ‌فیس[8] آورده شده است. مشخصات شبکه کم‌عمق نیز از طریق پیاده‌سازی و ارزیابی آن به‌دست‌آمده است. شبکه اصلی با مجموعه‌داده ImageNet آموزش دیده و بیش از یک میلیون پارامتر دارد. همچنین، حجم شبکه برابر با ۴/۷۳ مگابایت و عملیات ممیز شناور در ثانیه (FLOPs)[9] نیز درحدود ۰/۷۲ گیگ می‌باشد. درمقابل، شبکه کم‌عمق با مجموعه‌داده Cifar-10 آموزش‌دیده و بیش از ۷۲ هزار پارامتر دارد. حجم آن برابر ۰/۲۸ مگابایت بوده و FLOPs آن نیز درحدود ۶/۶۹ میلیون است. دقت شبکه اصلی و کم‌عمق به‌ترتیب برابر با ٪۸۰/۳ و ٪۸۵/۵۶ هستند. کلاس‌ها و ابعاد تصاویر موجود در مجموعه‌داده‌های ImageNet و Cifar-10 بایکدیگر متفاوت هستند و مقایسه نابرابری ایجاد می‌کنند. اما، به‌دلیل حجم بسیار بالای ImageNet و عدم دسترسی مستقیم به آن، پژوهش حاضر با مجموعه‌داده Cifar-10 انجام و حجم شبکه را کاهش داده است. بنابراین، با کاهش تعداد ماژول‌های Fire،

[1] Image Cache (ICache)
[2] Weights Cache (WCache)
[3] Multiply-Accumulate (MACC)
[4] Output Cache (OCache)
[5] Global Pooling Cache (GPoolCache)

[6] Synthesis
[7] github.com/forresti/SqueezeNet
[8] huggingface.co/qualcomm/SqueezeNet-1.1
[9] Floating Point Operations Per Second (FLOPs)

استفاده از Cifar-10 و افزایش دوره‌های آموزشی می‌توان بین تثبیت دقت، کاهش پارامترها و حجم شبکه تعادل برقرار کرد.



شکل (۱۱): بلوک دیاگرام طراحی سخت‌افزاری در محیط بلوکی Vivado

وزن‌ها و ساختار شبکه به ابزار Vitis HLS اعمال و پیاده‌سازی شتاب‌دهنده اجرا می‌شود. این‌کار در فرکانس ۱۰۰ مگاهرتز (زمان دوره تناوب ۱۰ نانوثانیه) انجام شده است لذا مدت زمان تأخیر[1] شبکه اصلی و کم‌عمق از رابطه (۱) تخمین زده شده و در جدول (٤) قرار گرفته‌اند. مطابق با نتایج به‌دست‌آمده حداقل سیکل‌ساعت مورد نیاز برای پردازش یک ورودی برای شبکه اصلی برابر با ۳۵۳ و برای شبکه کم‌عمق برابر با ١٦٥ است. مدت زمان دوره تناوبی که پس از پیاده‌سازی تخمین زده شده است برابر با ۹/۸۲ نانوثانیه (تقریباً برابر با مقدار دوره تناوب هدف یا ۱۰ نانوثانیه) می‌باشد. بنابراین، مدت زمان تأخیر برای شبکه‌های اصلی و کم‌عمق به‌ترتیب برابر با ٣٤٦٦/٤٦ و ١٦٢٠/٣ نانوثانیه هستند. این مدت زمان نشان می‌دهد که با کاهش پارامترها و حجم شبکه می‌توان سرعت پردازش تصاویر در شتاب‌دهنده را بهینه‌تر کرد.

$$\text{Latency(ns)} = \text{Latency (cycles)} \times \text{Clock Period (ns)} \quad (۱)$$

جدول (۳): مشخصات شبکه‌های SqueezeNet اصلی و کم‌عمق

اسم مدل	مجموعه‌داده	تعداد پارامترها	حجم مدل (MB)	FLOPs
SqueezeNet (8 Fire)	ImageNet	1.24M	4.73	0.72G
SqueezeNet (4 Fire)	Cifar-10	72.638K	0.28	6.69M

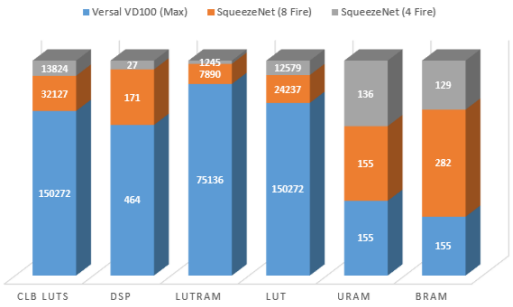
جدول (٤): میزان تأخیر شبکه‌های مورد استفاده

اسم مدل	Estimated (ns)	Latency (Minimum cycles)	Latency (ns)
SqueezeNet (8 Fire)	9.82	353	3466.46
SqueezeNet (4 Fire)	9.82	165	1620.3

[1] Latency

پارامترهای شبکه در این پژوهش (برخلاف ZynqNet) بین بلوک‌های BRAM و URAM پلتفرم Versal VD100 تقسیم و باعث کاهش تأخیر دسترسی به‌حافظه اصلی در زمان اجرای شتاب‌دهنده شده‌اند. به‌طورکلی، پلتفرم Versal VD100 دارای ١٥٥ واحد بلوک‌های BRAM (٥/٤ مگابایت) و URAM (٤٣/٦ مگابایت) است [١٠]. پلتفرم Zynq-7000 (XC7Z045) [٢٣] نیز تنها دارای ٥٤٥ واحد بلوک BRAM (١٩/٢ مگابایت) است که کمتر از مجموع حافظه‌های VD100 می‌باشد.

در این پژوهش با هدف مقایسه میزان مصرف حافظه‌های بلوکی شبکه SqueezeNet اصلی و کم‌عمق برروی تراشه پلتفرم پیاده‌سازی شده‌اند. جدول (٥) و شکل (١٢) میزان مصرف حافظه‌های داخلی توسط این دو شبکه را نشان می‌دهند. مشاهده می‌شود که شبکه اصلی از حافظه زیادی (٢٨٢ واحد BRAM و ١٥٥ واحد URAM) استفاده می‌کند. اما، منابع حافظه مصرفی شبکه کم‌عمق کمتر (١٢٩ واحد BRAM و ١٣٦ واحد URAM) می‌باشد. در جدول (٥) و شکل (١٢) میزان مصرف سایر منابع نیز آمده است و نشان می‌دهند که شبکه کم‌عمق کمتر از شبکه اصلی از بلوک‌های Versal VD100 استفاده می‌کند. افزایش تعداد FF ها در نسخه کم‌عمق ناشی از افزایش رجیسترهای کنترلی، سیگنال‌های زمان‌بندی و بافرهای میانی است که برای مدیریت جریان داده و کنترل حافظه مورد استفاده قرار گرفته‌اند و الزاماً با کاهش BRAM/URAM تناقض ندارد. برای شفافیت، مقادیر FF (به‌همراه LUT/DSP) در جدول (٧) گزارش شده و مبنای مقایسه "پس از پیاده‌سازی" است.



شکل (۱۲): نمودار منابع مصرفی

جدول (۵): مقایسه میزان مصرف حافظه‌های بلوکی

اسم مدل	BRAM	URAM	LUTRAM	LUT
SqueezeNet با هشت ماژول Fire	282	155	7.890K	24.237K
SqueezeNet با چهار ماژول Fire	129	136	1.245K	12.579K

جدول (٦) مقایسه توان مصرفی شبکه‌های اصلی و کم‌عمق را نشان می‌دهد. مدل کم‌عمق دارای مصرف کلی توان ۳/۹۸٤ وات می‌باشد که نشان‌دهنده طراحی کارآمدتری در مصرف انرژی است و آن‌را برای پیاده‌سازی‌های سبک و پایدار در دستگاه‌های با منابع محدود مناسب می‌سازد.

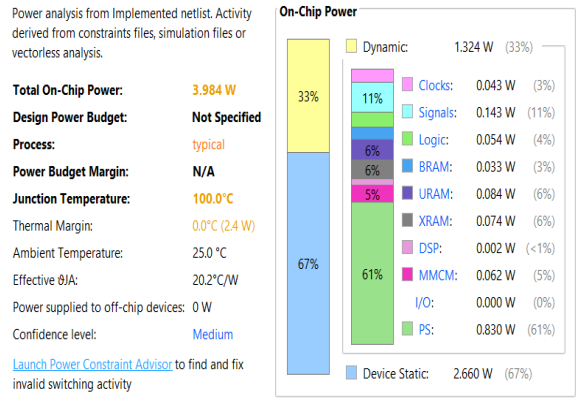
جدول (٦): مقایسه میزان توان مصرفی

اسم مدل	کل توان مصرفی (وات)	توان پویا (وات)	توان ثابت (وات)
SqueezeNet با هشت ماژول Fire	4.658	2.240	2.418
SqueezeNet با چهار ماژول Fire	3.984	1.324	2.660

شکل (۱۳) نیز گزارش دقیق‌تری از مصرف توان پلتفرم Versal VD100 در شتاب‌دهی شبکه کم‌عمق Squeezenet را ارائه می‌دهد. گزارش توان مصرفی با ابزار Vivado Power استخراج شده است. این گزارش مربوط به کل سیستم در دمای اتصال ۱۰۰ درجه سانتی‌گراد، فرکانس ۱۰۰ مگاهرتز و ولتاژ پیش‌فرض پلتفرم است. افزایش توان استاتیک عمدتاً ناشی از ویژگی‌های فناوری ساخت و شرایط حرارتی تراشه بوده و صرفاً به شتاب‌دهنده پیشنهادی محدود نمی‌شود.

بر اساس شکل (۱۳)، ٦۷٪ از توان کل (۲/٦٦ وات) مربوط به توان استاتیک و ۳۳٪ دیگر (۱/۳۲٤ وات) مربوط به توان دینامیک است.

توان استاتیک معمولاً ناشی از مصرف قطعات سازنده تراشه پلتفرم (ترانزیستورهای داخلی تراشه) در دمای بالای ۱۰۰ درجه سانتی‌گراد (دمای اتصال) می‌شود. توان دینامیک نیز شامل انرژی مصرفی در حین پردازش شبکه عصبی می‌شود. همان‌طورکه در شکل (۱۳) مشخص شده است ٦۱٪ توان برای پردازش نرم‌افزار و ۳۱٪ باقیمانده برای سایر بلوک‌های پلتفرم مصرف شده است. این پارامترها به معماری شبکه عصبی عمیق وابسته هستند. زیرا لایه‌های کانولوشن شبکه در حین پردازش منجر به فعالیت‌های الکتریکی (مثل تغییرات ولتاژ و جریان) در قطعات داخلی تراشه می‌شوند. لازم به‌ذکر است که دمای اتصال ۱۰۰ درجه سانتی‌گراد در کنار مقاومت حرارتی ۲۰/۲ درجه سانتی‌گراد بر وات، باعث ایجاد گرمای شدید در تراشه می‌شود. این امر نه‌تنها کارایی پردازش را کاهش می‌دهد بلکه عمر مفید تراشه را نیز تهدید می‌کند.



شکل (۱۳): گزارش توان مصرفی پلتفرم Versal VD100

در جدول (۷) نتایج پیاده‌سازی پژوهش‌های مختلف نشان داده شده است. پژوهشگران در [۳۳] از قابلیت بلوک DPU برای پیاده‌سازی شبکه‌عصبی CNN استفاده کرده‌اند. آن‌ها از طریق ابزار Vitis AI بلوک DPU را برای دو پلتفرم Versal و UltraScale استفاده کردند و به این نتیجه رسیدند که Versal عملکرد بهتری دارد. پژوهشگران در [۳٤] نیز سه شتاب‌دهنده BERT-Base، ViT-Base و BERT-Base(Limited AIE) را با چارچوب

شتاب‌دهنده ترانسفورماتور سفارشی (CAT)[1] برروی پلتفرم Versal VCK5000 پیاده‌سازی کرده و نتایج را به اشتراک گذاشته‌اند. پژوهشگران در [۳۹] شبکه AlexNet را روی پلتفرم ZCU104 اجرا کرده و پارامترهای شبکه را برروی حافظه‌های BRAM و URAM ذخیره کرده‌اند. شتاب‌دهنده ZynqNet [۲۲] توسط پیاده‌سازی شبکه SqueezeNet روی پلتفرم Zynq XC-7Z045 به‌وجود آمده است. این شتاب‌دهنده علی‌رغم محدودیت‌های منابعی که دارد، نتایج قابل قبولی را از خود نشان داده است. پژوهش حاضر الگوریتم شتاب‌دهنده ZynqNet را برروی Versal VD100 پیاده‌سازی کرده است با این تفاوت‌که حجم شبکه کاهش یافته و پارامترهای آن در حافظه‌های داخلی پلتفرم (BRAM و URAM) ذخیره شده‌اند. از آن‌جایی‌که ظرفیت حافظه‌های BRAM و URAM یکسان نیست لذا برای مقایسه منصفانه از معیار حافظه معادل BRAM یا $BRAM_{eq}$ استفاده می‌شود. این مقدار از رابطه (۲) و با فرض ظرفیت BRAM36 ≈ 36Kb و URAM ≈ 288Kb به‌دست می‌آید.

$$BRAM_{eq} = BRAM + 8 \times URAM \quad (۲)$$

. براین‌اساس، نتیجه‌گیری‌های مربوط به‌مصرف حافظه در جدول (۷) برمبنای BRAM معادل ($BRAM_{eq}$) گزارش می‌شوند و صرفاً تعداد بلوک‌های BRAM نیستند. لازم به تأکید است که در پیکربندی شامل ۸ ماژول Fire، تعداد پارامترهای شبکه تقریباً با نسخه مرجع SqueezeNet یکسان بوده و افزایش تعداد URAM ناشی از کمبود منابع BRAM است و ناشی از افزایش حجم پارامترها نیست. از آن‌جایی‌که به‌دلیل ملاحظات سرعت و معماری، امکان استفاده از بلوک‌های URAM برای ذخیره مجموعه‌های کوچک پارامتر وجود ندارد، لذا این بلوک‌های حافظه بدون بهره‌برداری کامل از ظرفیت خود تخصیص داده می‌شوند. بنابراین، بیان مصرف حافظه (صرفاً براساس تعداد URAM) می‌تواند باعث برآورد بیش‌ازحد میزان واقعی حافظه مصرفی شود. نتایج جدول (۷) نشان داد که شتاب‌دهی شبکه‌های عصبی با استفاده از پلتفرم‌های پیشرفته AMD Versal و استفاده حداکثری از امکانات آن‌ها می‌تواند میزان منابع مصرفی و تأخیر عملیاتی را کاهش دهد. با بهینه‌سازی اختصاصی این پلتفرم‌ها می‌توان مدل‌های پیچیده را نیز برروی آن‌ها پیاده‌سازی کرد.

۷- نتیجه‌گیری

این پژوهش، با هدف پیاده‌سازی شبکه SqueezeNet کم‌عمق (حاوی چهار ماژول Fire به‌جای هشت ماژول شبکه اصلی) برروی تراشه XCVE2302 پلتفرم AMD Versal VD100 انجام شده است. این کاهش ساختاری در عمق شبکه ضمن حفظ دقت (٪۸۵/۵۶) با استفاده از مجموعه‌داده CIFAR-10 باعث صرفه‌جویی در منابع سخت‌افزاری نیز شده است. برای اثبات صرفه‌جویی هر دو شبکه SqueezeNet اصلی و کم‌عمق با پلتفرم Versal VD100 پیاده‌سازی شد. نتایج نشان داد که BRAM مصرفی از ۲۸۲ واحد (شبکه اصلی) به‌مقدار ۱۲۹ واحد (شبکه کم‌عمق) و URAM مصرفی نیز از ۱۵۵ واحد به‌مقدار ۱۳۶ واحد کاهش یافته است. میزان مصرف LUT و LUTRAM نیز به‌ترتیب از ۲۴/۲۳۷ هزار به ۱۲/۵۷۹ هزار و از ۷/۸۹۰ هزار به ۱/۲۴۵ هزار رسیده است. پیاده‌سازی سخت‌افزاری در محیط Vitis HLS با بهره‌گیری از سلسله‌مراتب حافظه Versal پارامترهای شبکه را بین بلوک‌های URAM و BRAM تقسیم‌بندی کرده است. این رویکرد در پیاده‌سازی ZynqNet برروی تراشه XC7Z045 غیرممکن بود زیرا این پلتفرم از بلوک URAM پشتیبانی نمی‌کند. این نوع تقسیم‌بندی حافظه به‌همراه دسترسی موازی به کانال‌ها باعث کاهش تأخیر پردازش به‌مقدار ۱۶۲۰/۳ نانوثانیه (۰/۰۰۱۶۲ میلی‌ثانیه) می‌شوند. نتایج نشان می‌دهند که با طراحی شبکه کم‌حجم و مدیریت منابع حافظه می‌توان شتاب‌دهنده‌های کارآمدی را برروی دستگاه‌های قابل‌حمل پیاده‌سازی کرد.

این پژوهش به‌عنوان یک پایه شفاف و قابل کنترل در سطح HLS برای بهره‌برداری از سلسله‌مراتب حافظه‌های پلتفرم‌های پیشرفته (مانند Versal VD100) بستری را فراهم می‌کند تا در گام‌های بعدی، از قابلیت‌های منحصربه‌فرد معماری آن‌ها به‌طور کامل استفاده شود. مهم‌ترین مسیرهای توسعه سیستم‌های آتی شامل مهاجرت محاسبات به AI Engines برای شتاب بخشیدن

[1] Customized Transformer Accelerator (CAT)

به‌هسته‌های پردازشی convolution و activation با بهره‌وری انرژی بسیار بالاتر، استفاده هدفمند از شبکه روی تراشه (NoC) برای مدیریت کارآمد جریان داده بین AI Engines، حافظه‌ها و بخش PL و توسعه یک چارچوب طراحی ناهمگن یکپارچه است که بتواند به‌صورت خودکار وظایف را بین واحدهای پردازشی مختلف (DPU،AIE،PL) توزیع کند. این توسعه‌ها امکان پیاده‌سازی شبکه‌های عمیق‌تر و پیچیده‌تر را با حفظ کارایی و مصرف توان مناسب برای کاربردهای بلادرنگ در حوزه‌های رباتیک، خودروهای خودران و سیستم‌های نهفته صنعتی فراهم خواهد کرد.

جدول (۷): بررسی پارامترهای پژوهش‌های مختلف

مرجع	شبکه عصبی	پلتفرم	BRAM	URAM	LUT	BRAMeq	DSP	FF	Latency (ms)
[۲۲]	SqueezeNet (ZynqNet)	Zynq XC-7Z045	996	---	154K	996	739	137K	8.997
[۳۳]	CNN Model	Versal VCK190 (DPU C32B3)	0	332	227420	2656	779	---	2.98
		ZCU104 (DPU B4096)	257	0	53540	257	562	---	4.60
[۳٤]	BERT Base	Versal VCK5000	940	360	232.3K	3820	---	290.5K	0.118
	ViT Base		706	412	261.4K	4002	---	262.4K	0.129
	BERT Base (Limited AIE)		320	0	48.4K	320	---	73.1K	0.398
[٤٠]	AlexNet	ZCU104	198	80	101.95K	838	696	127.57K	102.76
این پژوهش	SqueezeNet (8 Fire)	Versal VD100	282	155	24.23K	1522	171	56.3K	0.00346
	SqueezeNet (4 Fire)		129	136	12.50K	1217	27	22.37K	0.00162

References

[1] S. V. Mahadevkar et al., "A review on machine learning styles in computer vision—techniques and future directions," Ieee Access, vol. 10, pp. 107293-107329, 2022, doi: 10.1109/ACCESS.2022.3209825.

[2] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," Proceedings of the IEEE, vol. 105, no. 12, pp. 2295-2329, 2017, doi: 10.1109/JPROC.2017.2761740.

[3] J. Lee, S. Kang, J. Lee, D. Shin, D. Han, and H.-J. Yoo, "The hardware and algorithm co-design for energy-efficient DNN processor on edge/mobile devices," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 10, pp. 3458-3470, 2020, doi: 10.1109/TCSI.2020.3021397.

[4] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," arXiv preprint arXiv:1602.07360, 2016, doi: 10.48550/arXiv.1602.07360.

[5] B. Koonce, "SqueezeNet," in Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization. Berkeley, CA: Apress, 2021, pp. 73-85.

[6] S. Mittal, P. Rajput, and S. Subramoney, "A survey of deep learning on CPUs: Opportunities and co-optimizations," IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 10, pp. 5095-5115, 2021, doi: 10.1109/TNNLS.2021.3071762.

[7] W. Jeon, G. Ko, J. Lee, H. Lee, D. Ha, and W. W. Ro, "Deep learning with GPUs," in Advances in computers, vol. 122: Elsevier, 2021, pp. 167-215.

[8] Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: A review," ieee Access, vol. 7, pp. 7823-7859, 2018, doi: 10.1109/ACCESS.2018.2890150.

[9] M. Hussain, "Sustainable machine vision for industry 4.0: a comprehensive review of convolutional neural networks and hardware accelerators in computer vision," AI, vol. 5, no. 3, 2024, doi: 10.3390/ai5030064.

[10] Versal AI Edge Series Development Board User Manual VD100, ALINX ELECTRONIC LIMITED, 2024. [Online]. Available:

https://www.alinx.com/public/upload/file/VD100_User_Manual_REV1.0.pdf.
[11] "The CIFAR-10 dataset." https://www.cs.toronto.edu/~kriz/cifar.html (accessed 2025).
[12] Y. Meng, C. Yang, S. Xiang, J. Wang, K. Mei, and L. Geng, "An efficient CNN accelerator achieving high PE utilization using a dense-/sparse-aware redundancy reduction method and data–index decoupling workflow," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 10, pp. 1537-1550, 2023, doi: 10.1109/TVLSI.2023.3298509.
[13] S. H. Hozhabr and R. Giorgi, "A Survey on Real-Time Object Detection on FPGAs," IEEE Access, 2025, doi: 10.1109/ACCESS.2025.3544515.
[14] L. Gao, Z. Luo, and L. Wang, "Convolutional Neural Network Acceleration Techniques Based on FPGA Platforms: Principles, Methods, and Challenges," Information, vol. 16, no. 10, p. 914, 2025, doi: 10.3390/info16100914.
[15] G. C. Marinó, A. Petrini, D. Malchiodi, and M. Frasca, "Deep neural networks compression: A comparative survey and choice recommendations," Neurocomputing, vol. 520, pp. 152-170, 2023, doi: 10.1016/j.neucom.2022.11.072.
[16] K. Vineetha, M. M. S. Reddy, C. Ramesh, and D. G. Kurup, "An efficient design methodology to speed up the FPGA implementation of artificial neural networks," Engineering Science and Technology, an International Journal, vol. 47, p. 101542, 2023, doi: 10.1016/j.jestch.2023.101542.
[17] Z. Li et al., "A high-performance pixel-level fully pipelined hardware accelerator for neural networks," IEEE Transactions on Neural Networks and Learning Systems, 2024, doi: 10.1109/TNNLS.2024.3423664.
[18] V. H. Kim and K. K. Choi, "A reconfigurable CNN-based accelerator design for fast and energy-efficient object detection system on mobile FPGA," IEEE Access, vol. 11, pp. 59438-59445, 2023, doi: 10.1109/ACCESS.2023.3285279.
[19] T. Sledevič, A. Serackis, and D. Plonis, "FPGA implementation of a convolutional neural network and its application for pollen detection upon entrance to the beehive," Agriculture, vol. 12, no. 11, p. 1849, 2022, doi: 10.3390/agriculture12111849.
[20] G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017, doi: 10.48550/arXiv.1704.04861.
[21] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6848-6856, doi: 10.48550/arXiv.1707.01083.
[22] D. Gschwend, "Zynqnet: An fpga-accelerated embedded convolutional neural network," arXiv preprint arXiv:2005.06892, 2020, doi: 10.48550/arXiv.2005.06892.
[23] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, The Zynq book: embedded processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 all programmable SoC. Strathclyde Academic Media, 2014.
[24] P. G. Mousouliotis and L. P. Petrou, "Squeezejet: High-level synthesis accelerator design for deep convolutional neural networks," in International Symposium on Applied Reconfigurable Computing, 2018: Springer, pp. 55-66, doi: 10.48550/arXiv.1805.08695.
[25] P. Mousouliotis, N. Tampouratzis, and I. Papaefstathiou, "SqueezeJet-3: An HLS-based accelerator for edge CNN applications on SoC FPGAs," in 2023 XXIX International Conference on Information, Communication and Automation Technologies (ICAT), 2023: IEEE, pp. 1-6, doi: 10.1109/ICAT57854.2023.10171329.
[26] Z. Zhao et al., "A High-Throughput FPGA Accelerator for Lightweight CNNs With Balanced Dataflow," IEEE Transactions on Circuits and Systems I: Regular Papers, 2025, doi: 10.48550/arXiv.2407.19449.
[27] L. Zhou, "Design and Implementation of Embedded Image Processing Chip Based on Lightweight MobileNetV3," in 2025 6th International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE), 2025: IEEE, pp. 309-313, doi: 10.1109/ICBASE66587.2025.11181362.
[28] R. Yarnell, M. Hossain, and R. F. DeMara, "Image Quantization Tradeoffs in a YOLO-based FPGA Accelerator Framework," in 2023 24th International Symposium on Quality Electronic Design (ISQED), 2023: IEEE, pp. 1-7, doi: 10.1109/ISQED57927.2023.10129324.
[29] A. Parameshwara and S. H. Mokashi, "FPGA-Accelerated RISC-V ISA Extensions for Efficient Neural Network Inference on Edge Devices," arXiv preprint arXiv:2511.06955, 2025, doi: 10.48550/arXiv.2511.06955.
[30] H. Yan and Y. Ma, "A reconfigurable heterogeneous computing-hardware design with Aimed VGG16 acceleration," in 2021 IEEE

International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), 2021: IEEE, pp. 274-278, doi: 10.1109/CEI52496.2021.9574533.
[31] "AMD Versal AI Edge." https://www.amd.com/en/products/adaptive-socs-and-fpgas/versal/ai-edge-series.html (accessed 2025).
[32] T. D. S. Ramos, "Implementation of Convolutional Neural Networks on a Versal Device," Universidade do Porto (Portugal), 2023.
[33] A. Al-Zoubi, G. Martino, F. H. Bahnsen, J. Zhu, H. Schlarb, and G. Fey, "Cnn implementation and analysis on xilinx versal acap at european xfel," in 2022 IEEE 35th International System-on-Chip Conference (SOCC), 2022: IEEE, pp. 1-6, doi: 10.1109/SOCC56010.2022.9908101.
[34] W. Zhang, Y. Liu, and Z. Bao, "Cat: Customized transformer accelerator framework on versal acap," arXiv preprint arXiv:2409.09689, 2024, doi: 10.48550/arXiv.2409.09689.
[35] N. Perryman, C. Wilson, and A. George, "Evaluation of xilinx versal architecture for next-gen edge computing in space," in 2023 IEEE aerospace conference, 2023: IEEE, pp. 1-11, doi: 10.1109/AERO55745.2023.10115906.
[36] T. Knopp, J. Chu, and S. Ahmad, "AMD versal™ AI edge series gen 2 for vision and automotive," in 2024 IEEE Hot Chips 36 Symposium (HCS), 2024: IEEE Computer Society, pp. 1-28, doi: 10.1109/hcs61935.2024.10665274.
[37] P. Dong et al., "EQ-ViT: Algorithm-hardware co-design for end-to-end acceleration of real-time vision transformer inference on Versal ACAP architecture," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 43, no. 11, pp. 3949-3960, 2024, doi: 10.1109/tcad.2024.3443692.
[38] Z. Bao, T. Zang, Y. Liu, and W. Zhang, "Efficient Number Theoretic Transform accelerator on the versal platform powered by the AI Engine," Future Generation Computer Systems, vol. 166, p. 107728, 2025, doi: 10.1016/j.future.2025.107728.
[39] M. Petry, G. Wuwer, A. Koch, P. Gest, M. Ghiglione, and M. Werner, "Accelerated deep-learning inference on the versal adaptive SoC in the space domain," in 2023 European Data Handling & Data Processing Conference (EDHPC), 2023: IEEE, pp. 1-8, doi: 10.23919/EDHPC59100.2023.10396011.
[40] M. Zhang, L. Li, H. Wang, Y. Liu, H. Qin, and W. Zhao, "Optimized compression for implementing convolutional neural networks on FPGA," Electronics, vol. 8, no. 3, p. 295, 2019, doi: 10.3390/electronics8030295.

Journal of Applied And Basic Machine Intelligence Research (ABMIR)

Applied and Basic Machine Intelligence Research

Vol. 4, No. 1, Spring & Summer 2026, pp. 113 - 128

A Lightweight Implementation of a Reduced-Depth SqueezeNet CNN on the Versal Platform for Edge Computing Applications

Zahra Ghasemi[1], Atefeh Salimi Shahraki[1*], Mahdi Abbasi[2,3], Mohammad Lali Dastjerdi[1]

[1]Department of Electrical Engineering, Isf.C., Islamic Azad University, Isfahan, Iran

[2]Department of Computer Engineering, Engineering Faculty, Bu-Ali Sina University Hamedan, Iran

[3]School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

Article Information

Original Research Paper

Received:
2025 November 26

Accepted:
2026 February 12

Keywords:
Convolutional Neural Network, SqueezeNet, Versal ACAP, FPGA Accelerator, Edge AI

Corresponding Author*:
Atefehsalimi@iau.ac.ir

Abstract

In recent years, deep neural networks have been widely adopted in computer vision applications. However, deploying these networks on portable and edge devices faces serious challenges due to limited computational resources, memory capacity, and power consumption constraints. One effective approach to overcoming these limitations is the use of advanced heterogeneous platforms such as AMD Versal. In this work, the implementation of a shallow version of the SqueezeNet network, consisting of four Fire modules, on the XCVE2302 device of the AMD Versal VD100 platform is investigated. First, the network was implemented using the PyTorch library in the Google Colab environment and trained on the CIFAR-10 dataset. Subsequently, the network architecture along with the trained weights was mapped to hardware using the Vitis HLS tool within Vivado 2023. The final implementation operates at a clock frequency of 100 MHz and utilizes 129 BRAM blocks and 136 URAM blocks, with a total power consumption of approximately 3.984 W. The obtained results demonstrate that by designing lightweight neural networks and employing efficient memory resource management, it is possible to implement efficient accelerators on the Versal platform. Furthermore, fully exploiting the architectural capabilities of AMD Versal can significantly reduce the challenges associated with deploying deep neural networks on hardware accelerators.

: 10.22034/ABMIR.2026.24030.1191

E-ISSN: 2821-2037